Lab Exercise 4– Terraform Variables

Objective:

Learn how to define and use variables in Terraform configuration.

Prerequisites:

- Install Terraform on your machine.

Steps:

1. Create a Terraform Directory:

- Create a new directory for your Terraform project.

```
mkdir terraform-variables

cd terraform-variables
```

2. Create a Terraform Configuration File:

- Create a file named main.tf within your project directory.

 # main.tf

```
variable "ami" {
    description = "AMI ID"
default = "ami-03f4878755434977f"
}

variable "instance_type" {
    description = "EC2 Instance Type"
    default     = "t2.micro"
}
```

main.tf > provider "aws" > region

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}
provider "aws" {
region = "var.region"
access_key = "AKIAZW6RGWG6KYNVNEET"
secret_key = "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"
}
```

3.Define Variables:

- Open a new file named variables.tf. Define variables for region, ami, and instance_type.

 # variables.tf

main.tf | variable.tf | instance.tf

variable.tf > ...

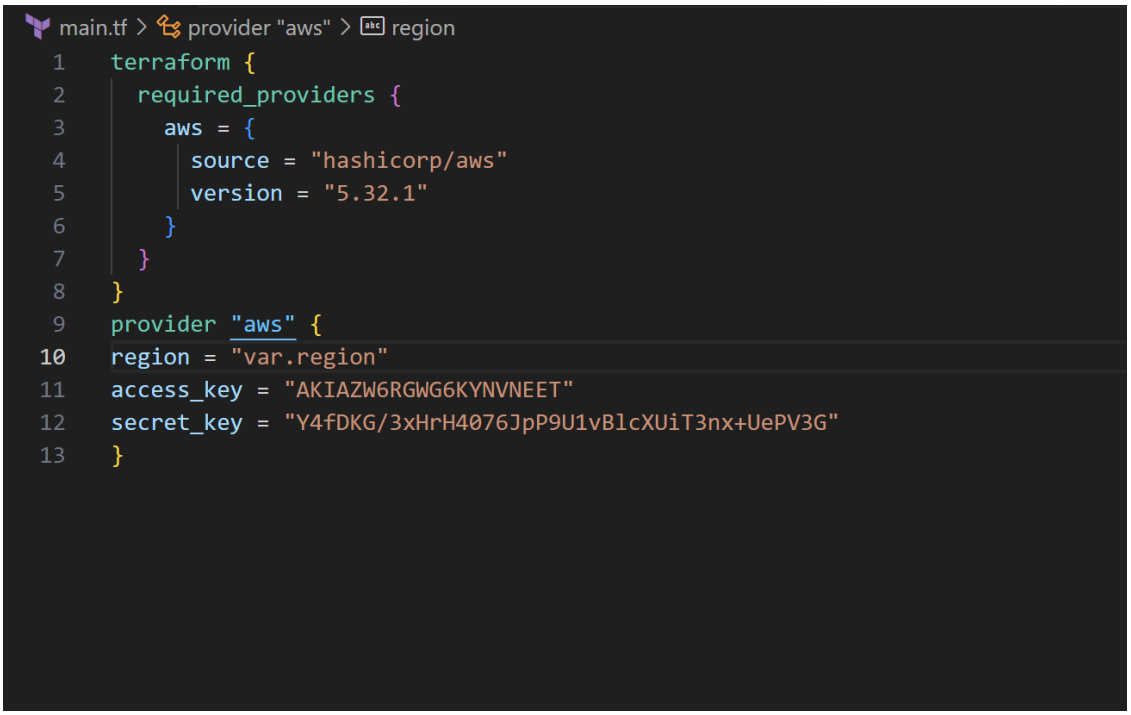
```
variable "region" {
    description = "AWS region"
    default = "ap-south-1"
}


variable "ami" {
    description = "AMI ID"
default = "ami-03f4878755434977f"
}
```

4.Use Variables in main.tf:

- Modify main.tf to use the variables.

 # main.tf



5.Initialize and Apply:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS E:\terraform-variables>
```

- Run the following Terraform commands to initialize and apply the configuration.

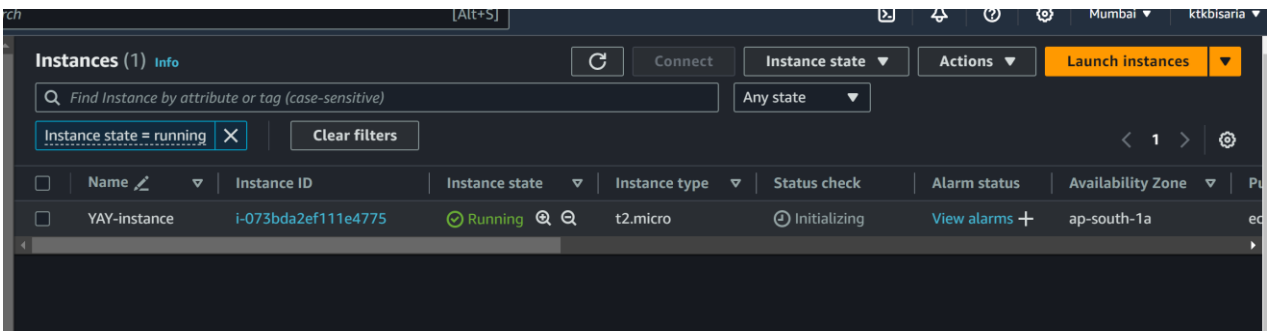
```
PS E:\terraform-variables> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions a
  + create

Terraform will perform the following actions:

  # aws_instance.YAYinstance[0] will be created
  + resource "aws_instance" "YAYinstance" {
      + ami                                  = "ami-03f4878755434977f"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
```

Observe how the region changes based on the variable override.



6.Clean Up:

After testing, you can clean up resources.

```
Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.YAYinstance[0]: Destroying... [id=i-073bda2ef111e4775]
aws_instance.YAYinstance[0]: Still destroying... [id=i-073bda2ef111e4775, 10s elapsed]
aws_instance.YAYinstance[0]: Still destroying... [id=i-073bda2ef111e4775, 20s elapsed]
aws_instance.YAYinstance[0]: Still destroying... [id=i-073bda2ef111e4775, 30s elapsed]
aws_instance.YAYinstance[0]: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
PS E:\terraform-variables>
```

Confirm the destruction by typing yes.